Rapport de stage

Mise en place d'un environnement de qualification pour la plateforme PLM

Pierric Grguric

Année 2014–2015

Stage de 2e année réalisé au LORIA

en vue de la validation de la 2e année d'études à TELECOM Nancy

Maître de stage : Martin Quinson, Gérald Oster

Encadrant universitaire : Suzanne Collin

Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : Grguric, Pierric

Élève-ingénieur(e) régulièrement inscrit(e) en 2e année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31314760

Année universitaire : 2014–2015

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

Mise en place d'un environnement de qualification pour la plateforme PLM dediée à l'apprentissage de la programmation

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Vandoeuvre-lès-Nancy, le 25 août 2015

Signature :

Rapport de stage

Mise en place d'un environnement de qualification pour la plateforme PLM

Pierric Grguric

Année 2014–2015

Stage de 2e année réalisé au LORIA

en vue de la validation de la 2e année d'études à TELECOM Nancy

Pierric Grguric
8, rue Jacques Callot
54500, Vandoeuvre-lès-Nancy
+33 (0)6 02 36 08 35
pierric.grguric@telecomnancy.net

TELECOM Nancy
193 avenue Paul Muller,
CS 90172, VILLERS-LÈS-NANCY
+33 (0)3 83 68 26 00
contact@telecomnancy.eu

LORIA
Campus scientifique
54506, Vandoeuvre-lès-Nancy
+33 (0)3 83 59 20 00

Maître de stage : Martin Quinson, Gérald Oster

Encadrant universitaire : Suzanne Collin

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé tout au long de mon stage.

Plus particulièrement, je tiens à remercier mes maîtres de stage MM. Martin Quinson et Gérald Oster pour l'encadrement et les conseils qu'ils m'ont apporté pendant mon stage.

Je remercie également M. Mathieu Nicolas pour son aide, notamment pour la compréhension du code existant.

Table des matières

1 Introduction

La "Programmer's Learning Machine" (PLM) est un logiciel d'apprentissage des bases de la programmation informatique. Jusqu'à maintenant, il n'existait qu'une version "locale" de la PLM, ce qui posait un problème en termes de possibilité de diffusion. Pour remédier à ce problème, la décision a été prise de transformer le logiciel en une application web. L'un des inconvénients de cette solution est le partage des ressources.

C'est dans le cadre de cette décision que s'inscrit mon stage. Plus particulièrement, j'ai été chargé de mettre en place un environnement de test pour déterminer la charge que peut supporter cette nouvelle version de la PLM.

De prime abord, nous verrons plus en détail le contexte du stage, à savoir l'établissement d'accueil : le LORIA, ainsi que le logiciel PLM.

Ensuite, nous verrons le déroulement de mon stage, c'est-à-dire les objectifs et la manière dont je les aient complétés.

Pour finir, nous dresserons le bilan de ce stage.

2 Présentation

2.1 Présentation du LORIA

Le LORIA (laboratoire lorrain de recherche en informatique et ses applications) est une Unité Mixte de Recherche (UMR 7503), commune à plusieurs établissements : le CNRS (Centre National de la Recherche Scientifique), l'Université de Lorraine et Inria (Institut National de Recherche en Informatique et Automatique). Depuis sa création en 1977, sa mission est la recherche fondamentale et appliquée en sciences informatiques.

le LORIA emploi 450 personnes réparties en cinq départements (Figure 2.1) [1] :

- Algorithmique, calcul, image et géométrie (6 équipes)
- Méthodes formelles (6 équipes)
- Réseaux, systèmes et services (3 équipes)
- Traitement automatique des langues et des connaissances (8 équipes)
- Systèmes complexes et intelligence artificielle (5 équipes)

J'ai été accueilli dans l'équipe VERIDIS (dont la mission principale est la conception de systèmes distribués et la vérification des systèmes) qui fait parti du département Méthodes formelles.

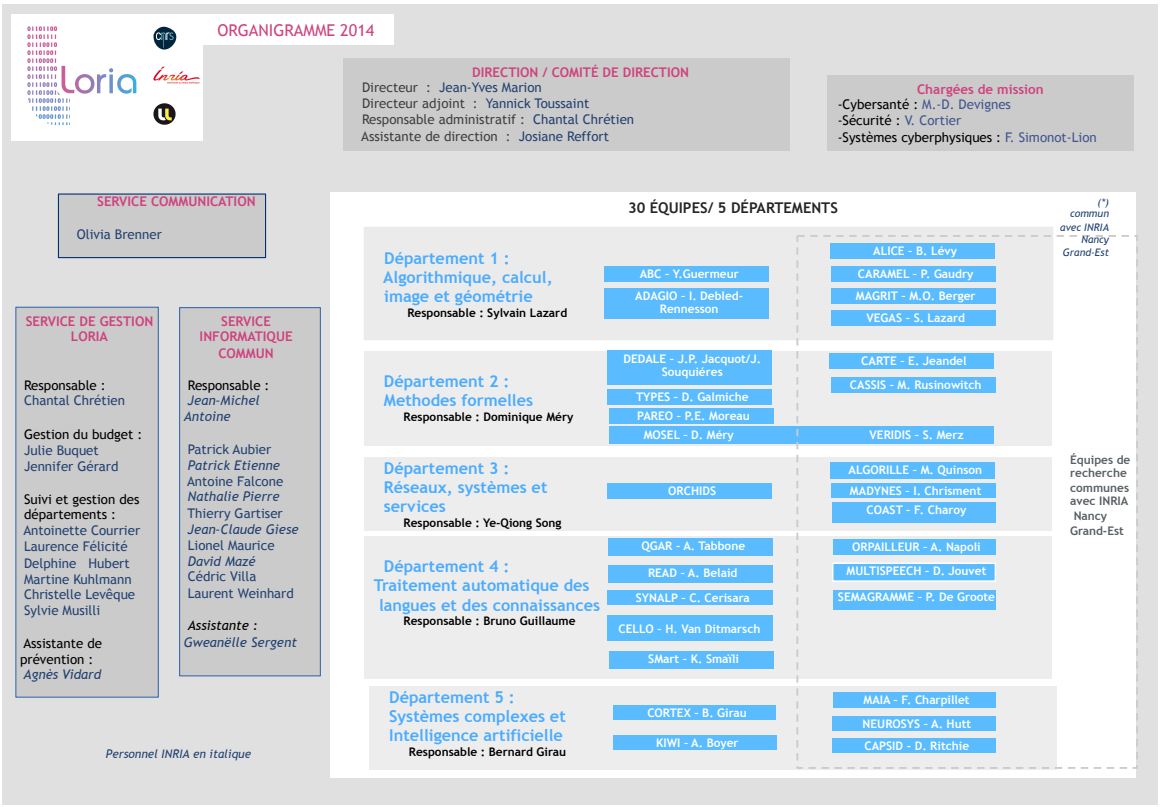

FIGURE 2.1 – Organigramme du LORIA

2.2 Présentation de la PLM

MM. Martin Quinson et Gérald Oster ont démarré le projet PLM en 2008 dans le but de faciliter les débuts dans le monde de la programmation aux élèves de TELECOM Nancy. La PLM permet d'apprendre les bases de la programmation dans plusieurs langages (Java, Python, Scala) à travers 200 exercices allant de la déclaration de variable à l'écriture de fonction récursives.[2]

2.2.1 PLM version lourde

A l'origine, la PLM a été développée en tant qu'application Java avec interface Swing, ce qui implique que l'utilisateur devait lancer le programme sur son ordinateur, après avoir préalablement téléchargé l'archive jar de la PLM. L'un des inconvénient était que la progression de l'utilisateur devait être exportée manuellement lorsque l'on souhaitait changer de machine.

Une fois le logiciel lancé, l'utilisateur a accès à différents univers regroupant plusieurs exercices, classés de sorte à assurer une progression logique de l'utilisateur. Le premier univers a pour objectif de faire évoluer des "buggles" (petite bêtes issu du jeu space invaders) dans différents mondes, ce qui a pour effet d'apporter un coté récréatif aux premiers niveaux, et donc l'utilisateur ne se sent pas perdu dans du "code pur".

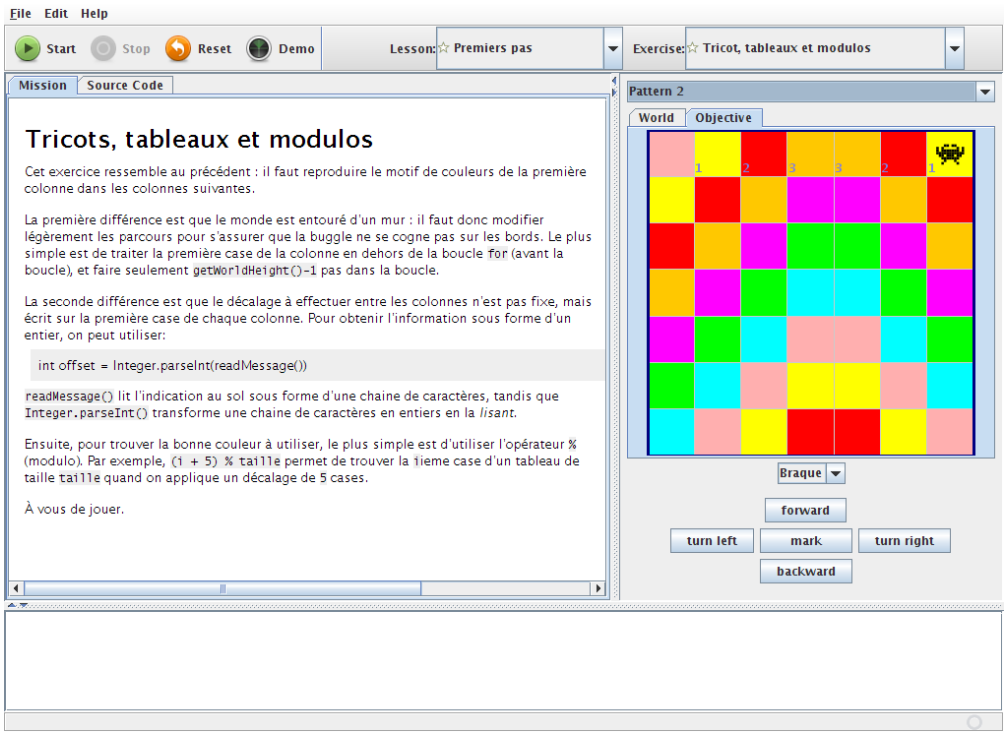

FIGURE 2.2 – Apercu de la PLM version lourde

2.2.2 PLM version web

Pour la rentrée scolaire 2015, l'objectif du projet PLM était de se débarrasser du client lourd et de passer à une version web de la PLM, ceci afin de supprimer l'étape de téléchargement, d'avoir la possibilité de continuer son code facilement même si l'on change de machine, et de

pouvoir diffuser la PLM à une plus large échelle, puisqu'il suffira d'accéder à une page web pour commencer à utiliser la PLM.

Cette nouvelle version a entraîné un changement de technologie, puisque le rendu graphique, auparavant réalisé en Swing, est maintenant assuré par AngularJS[3], ainsi que canvas pour le rendu des mondes. Un serveur serveur web tournant sous Play Framework a également du être mis en place, afin d'exécuter le code des utilisateurs. La connexion entre les utilisateurs (clients) et le serveur se fait à l'aide d'une WebSocket, ce qui permet un échange constant.

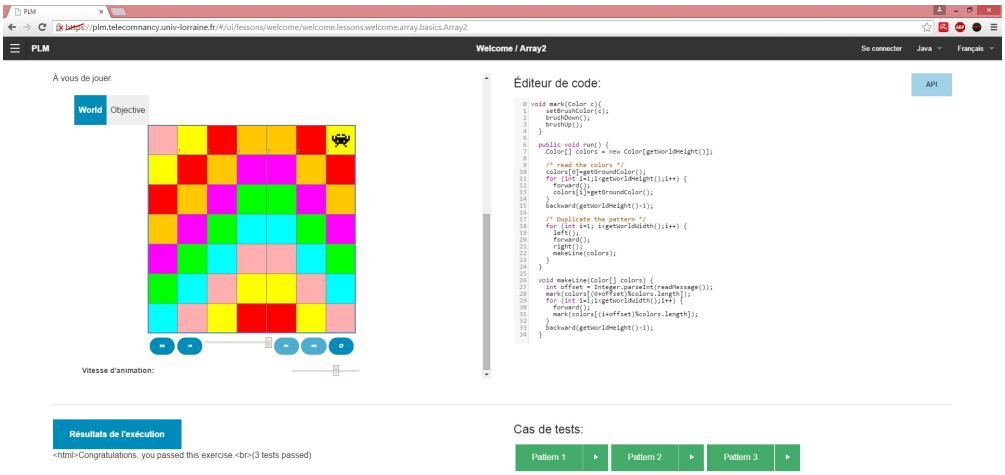

FIGURE 2.3 – Apercu de la PLM version web

3 Déroulement du stage

C'est dans ce contexte de changement de version que mon stage c'est déroulé. En effet, les technologies utilisées n'étant plus les mêmes, il était nécessaire que certaines parties de la PLM soient recodées dans un autre langage. Afin de s'assurer que ces changements ne modifient pas le comportement de la PLM, il a été envisagé de mettre en place un environnement de test pour la PLM. C'est là l'objectif de mon stage. Il s'est découpé en trois grandes parties à savoir : établir un script de lancement afin de déployer rapidement un serveur, écrire des tests unitaires afin de s'assurer que les composantes de la PLM se comportent comme attendu, et écrire des tests E2E afin de simuler le parcours d'un utilisateur.

3.1 Infrastructure Docker

La première partie du stage était d'avoir un script de lancement de la PLM. En effet, les tests étant susceptible de faire crasher le serveur, il était nécessaire d'avoir un script permettant de le relancer rapidement.

Pour ce faire, j'ai choisi d'utiliser la technologie docker qui permet d'avoir une image équivalente à une machine virtuelle possédant uniquement les composantes souhaitées. Une image docker se créée de la façon suivante : on écrit un dockerfile qui liste l'ensemble des commandes nécessaires à l'obtention de l'état dans lequel on souhaite avoir notre image docker. Ainsi la "compilation" du dockerfile génère une image docker qui est dans l'état que l'on souhaite, "prête à l'emploi" en quelque sorte. L'avantage de cette technologie est qu'une fois la compilation effectuée la première fois, il n'est plus nécessaire de la relancer et donc on peut démarrer un serveur local hébergeant la PLM en moins d'une minute, en une seule ligne de commande (comparé à une douzaine de minutes pour un démarrage "manuel")

Un des avantages de la technologie docker est que les dockerfiles peuvent hériter d'autres dockerfiles, ainsi on peut repartir d'une image docker déjà existante et exécuter quelques lignes de commandes à l'intérieur de cette image afin de créer une image docker plus particulière. J'ai donc profité de cet avantage en découpant le script en deux parties :
-La première partant d'une image sous ubuntu avec un jdk ou je rajoute le package PlayFramework. C'est la partie "stable" du script.
-La deuxième partie repart de cette image docker pour y télécharger la dernière version de PLM et la lancer. Ainsi en cas de modification de la PLM, il n'y a que la compilation du deuxième dockerfile à relancer. Ce script est présenté en Première annexe.

3.2 Test unitaire

La deuxième partie du stage consistait en l'écriture de tests unitaires qui devait s'assurer que les différentes composantes des univers de la PLM se comportaient comme attendu.

Il existait déjà des tests sur l'univers Buggle. Ces tests sont écrit en node.js. Après une étape de compréhension des tests existant, j'ai donc écrit les tests pour les autres univers dans le même langage. Un exemple de test est présenté en Deuxième annexe.

Le but de ces tests est de s'assurer que les fonctions qui manipulent les différents objets présents à l'intérieur de PLM modifient correctement ces objets lorsqu'elles sont appelées. L'écriture de ces tests m'a notamment permis de détecter une erreur d'affichage dans le monde des pancakes, que j'ai pu corriger.

3.3 Test EndToEnd

La troisième et dernière partie du stage était consacrée à l'écriture de tests EndToEnd (E2E), c'est-à-dire de tests allant d'un bout à l'autre de l'application. Le but de ce type de test est de simuler le comportement d'un utilisateur navigant à travers l'application.

La PLM utilise Angular.js comme librairie principale pour l'interface graphique. Par conséquent, j'ai décidé d'utiliser protractor qui est un framework de test E2E pour les applications utilisant angular.js. L'avantage de protractor est qu'il se synchronise avec la page actuellement testée, ce qui fait que l'on n'a en théorie pas besoin de gérer les latences dû au serveur ou au navigateur. Malheureusement la PLM utilise une websocket pour réaliser les échanges entre le client et le serveur, ce qui fait que protractor ne peut plus se synchroniser (à cause des requêtes permanentes). J'ai donc du gérer manuellement la synchronisation entre protractor et la page web qui était testée.

Un autre avantage de protractor est que les tests écrit sont facilement scalable. Ainsi, les tests écrits pourront être réutilisé lors de la conception de tests de charges. Il était prévu que j'en établissent dans le cadre de mon stage, mais par manque de temps je n'ai pas pu en effectuer. Un exemple de test protractor est présenté en Troisième annexe.

4 Bilan

Les tests écrits pendant mon stage permettent actuellement d'assurer un déploiement rapide du serveur, et garantissent une fiabilité dans le comportement interne de l'application, ainsi que le comportement global de la PLM vis à vis de l'utilisateur.

L'une des améliorations possibles de la partie test de la PLM est la mise en place de tests précis indiquant les ressources nécessaires pour supporter un nombre d'utilisateurs donné.

Actuellement, la PLM n'est utilisé que dans le cadre de la formation d'ingénieur dispensé à TELECOM Nancy. Le passage à la version web est une première étape dans la diffusion de la PLM à plus grande échelle, il est prévu de l'intégrer à un MOOC du Loria.

5 Conclusion

Lors de ce stage, j'ai pris conscience de l'importance et de l'ampleur que peuvent avoir les tests. J'ai pu renforcer mes connaissances sur les scripts et les tests ainsi que découvrir les outils relatifs (docker, protractor). Ce stage m'a également confronté à l'expérience de compréhension d'un code déjà existant afin de le tester.

Ce stage c'est déroulé dans un laboratoire de recherche, ce qui m'a permis d'avoir un aperçu du monde de la recherche. J'ai eu l'occasion de contribuer à un projet en cours ce qui a été l'occasion pour moi de travailler en collaboration avec d'autres personnes. Ainsi, ce stage fût pour moi une expérience enrichissante aussi bien sur le plan personnel que professionnel.

Bibliographie / Webographie

[1] http://www.loria.fr/le-loria-1/organisation/organigramme. 3
[2] http://www.loria.fr/~quinson/Teaching/PLM/. 5
[3] https://angularjs.org/. 6

Liste des illustrations

Glossaire

- E2E : EndToEnd
- jdk : java development kit
- LORIA : laboratoire lorrain de recherche en informatique et ses applications
- PLM : Programmer's Learning Machine

Annexes

A Première Annexe

exemple de dockerfile :

```
FROM pierricgrguric/docker-play-framework

RUN apt-get -y update && apt-get -y install git
RUN git clone https://github.com/MatthieuNICOLAS/webPLM.git --
   depth 1
WORKDIR webPLM
RUN activator compile
RUN activator clean stage

CMD ["./target/universal/stage/bin/web-plm"]
```

B Seconde Annexe

exemple de test unitaire :

```
(function(){
        'use_strict';

        describe('FlipOperation', function() {
        var _FlipOperation;

        var currentWorld;
        var flipOperation;
        var number;
        var pancakeStack;
        var moveCount;
        var numberFlip;

        beforeEach(module('PLMApp'));

        beforeEach(inject(function(FlipOperation) {
                        _FlipOperation = FlipOperation;
                }));

        beforeEach(function() {
            var i;
            var nbPancake;
            var memory;

            pancakeStack = [];
            nbPancake = getRandomInt(20) + 2;
            for(i = 0; i < nbPancake; i++) {
                pancakeStack.push({radius: i,upsideDown: true});
            }


            number = getRandomInt(nbPancake);
            moveCount = getRandomInt(42);
            numberFlip = getRandomInt(42);

            currentWorld = {
                pancakeStack: pancakeStack,
```

```
            moveCount: moveCount,
            numberFlip: numberFlip
        };

        var dataOperation = {
            number: number,
            oldNumber: numberFlip
                    };

        flipOperation = new _FlipOperation(dataOperation);
});

it('should be initialized correctly by its constructor',
    function () {
        expect(flipOperation.number).toEqual(number);
        expect(flipOperation.oldNumber).toEqual(numberFlip);
            });

            it('should flip pancake between number and the
                end when applied', function () {
        var length = currentWorld.pancakeStack.length;
                    flipOperation.apply(currentWorld);
        for (var i=1;i<=number;i++){
            expect(currentWorld.pancakeStack[length-i].radius
                ).toEqual(length-number+i-1);
            expect(currentWorld.pancakeStack[length-i].
                upsideDown).toEqual(false);
        }
        for (var i=number+1;i<=length;i++){
            expect(currentWorld.pancakeStack[length-i].radius
                ).toEqual(length-i);
            expect(currentWorld.pancakeStack[length-i].
                upsideDown).toEqual(true);
        }
        expect(currentWorld.moveCount).toEqual(moveCount+1);
        expect(currentWorld.numberFlip).toEqual(number);
            });

            it('should reflip pancake between number and the
                end when reversed', function () {
        var length = currentWorld.pancakeStack.length;
                    flipOperation.reverse(currentWorld);
        for (var i=1;i<=number;i++){
            expect(currentWorld.pancakeStack[length-i].radius
                ).toEqual(length-number+i-1);
            expect(currentWorld.pancakeStack[length-i].
                upsideDown).toEqual(false);
        }
        for (var i=number+1;i<=length;i++){
```

```
                expect(currentWorld.pancakeStack[length-i].radius
                    ).toEqual(length-i);
                expect(currentWorld.pancakeStack[length-i].
                    upsideDown).toEqual(true);
            }
            expect(currentWorld.moveCount).toEqual(moveCount-1);
            expect(currentWorld.numberFlip).toEqual(numberFlip);
                });

        it('should not change currentWorld when applied then
            reversed', function () {
                        var current = {
                pancakeStack: currentWorld.pancakeStack.slice(),
                moveCount: moveCount,
                numberFlip: numberFlip
            };
                        flipOperation.apply(currentWorld);
                        flipOperation.reverse(currentWorld);
                        expect(currentWorld).toEqual(current);
                });

        it('should not change currentWorld when reversed then
            applied', function () {
                        var current = {
                pancakeStack: currentWorld.pancakeStack.slice(),
                moveCount: moveCount,
                numberFlip: number
            };
                        flipOperation.reverse(currentWorld);
                        flipOperation.apply(currentWorld);
                        expect(currentWorld).toEqual(current);
                });
        });
})();
```

C Troisième Annexe

exemple de test protractor :

```
describe('Exo1', function() {

    var until = protractor.ExpectedConditions;
    var cross = element(by.css('.close-reveal-modal[ng-click]'));
    var button = element(by.css('.button-with-icon'));
    var redButton = element(by.css('ul:first-child li .button-
       with-icon.alert'));
    var congratsW = element(by.css('#successModal[style]'));
    var congratsTitle = element(by.css('#successModal h2 span'));
    var congratsCross = element.all(by.css('.close-reveal-modal')
       ).get(2);

    beforeEach(function() {
        browser.get('https://plm.telecomnancy.univ-lorraine.fr/#/
           ui/lessons/welcome/');
        browser.ignoreSynchronization=true;
        browser.wait(until.or(until.visibilityOf(cross),until.
           visibilityOf(button)), 5000, "Pop-up isn't here");
        cross.isDisplayed().then(function(isVisible){
            if (isVisible)
                browser.actions().click(cross).perform();
        });
    });

    afterEach(function() {
        browser.executeScript('window.sessionStorage.clear();');
        browser.executeScript('window.localStorage.clear();');
    });

    it('should warn the user that world are unequal if no code is
       submitted', function() {
        browser.wait(until.visibilityOf(button), 500, "Button
           unclickable");
        button.click();
        browser.wait(until.visibilityOf(redButton), 5000, "Button
           isn't red");
        expect(protractor.ExpectedConditions.visibilityOf(
           redButton)()).toEqual(true);
```

```
    });

    it('should congrats the user for passing the exercise',
       function() {
        browser.executeScript("window.editor.setValue(\"avance()
           ;\"); ");
        browser.wait(until.visibilityOf(button), 5000, "Button
           unclickable");
        button.click();
        browser.wait(until.visibilityOf(congratsW), 5000, "
           Congrats pop-up isn't here");
        browser.wait(until.textToBePresentInElement(congratsTitle
           ,'Exercice reussi'), 5000, "Congrats pop-up isn't here
           ");
        expect(congratsTitle.getText()).toEqual('Exercice reussi'
           );
        congratsCross.click();
        browser.wait(until.visibilityOf(button), 500, "Pop-up
           still here");
    });
});
```

Résumé

L'objectif du projet PLM (Programmer's Learning Machine) est de proposer un outil permettant d'apprendre les bases de la programmation. Dans le cadre de son développement en version web, il a été nécessaire de tester certains aspect à savoir : des tests unitaires (pour le comportement interne) et des tests de bouts en bouts (simulation d'un utilisateur).

Mots-clés : Docker, test unitaire, test E2E

Abstract

The PLM (Programmer's Learning Machine) project's goal is to create a software which help beginners to learn to program. For the development of the web version, it has been mandatory to test some parts of the software. More precisely unit testing (for intern behaviour) and EndToEnd testing (simulation of an user).

Keywords : Docker, unitary testing, E2E testing